UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P O Box 1450
Alexandria, Virginia 22313-1450
www uspto gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/541,458	07/06/2005	Masaki Kawai	2005_1088A	8952

52349 7590 12/10/2008
WENDEROTH, LIND & PONACK L.L.P.
2033 K. STREET, NW
SUITE 800
WASHINGTON, DC 20006

EXAMINER	
KENDALL, CHUCK O	

ART UNIT	PAPER NUMBER
2192	

MAIL DATE	DELIVERY MODE
12/10/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/541,458	KAWAI ET AL.	
	Examiner	Art Unit	
	CHUCK O. KENDALL	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *06 July 2005*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-31* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-31* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)	4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)	5) ☐ Notice of Informal Patent Application
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *7/05,08/05,0507*.	6) ☐ Other: _____.

DETAILED ACTION

1. This is in response to application filed 07/06/05.

2. Claims 1 - 31 have been examined and stand rejected.

Claim Rejections - 35 USC § 101

3. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

4. Claims 1 – 21, 23 – 25, and 27 – 31 is directed to non-statutory subject matter because the claimed invention appears have claim limitations directed to software per se type limitations (e.g. compiler program and object program), and without the associated hardware tied in the claims with the software limitations it doesn't appear that the claims will be able to carry out it's intended function.

Claim Rejections - 35 USC § 102

5. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

6. Claims 1 – 20, 22 – 31 are rejected under 35 U.S.C. 102(b) as being anticipated by Bates 6,049,6657.

Regarding claims 1, 8, and 22 - 28 a compiler program for converting a source program into an object program and causing a computer to function as:

an address saving program generating means for generating an address saving program for saving a data memory area address used by a calling program module included in the source program (FIG. 3, 106 and all associated text),

an address setting program generating means for generating an address setting program for setting a data memory area address used by an other program module to be called by the calling program module (6:37 - 43, see setting pointer),

a transferring program generating means for generating a transferring program for the transfer from a first subprogram included in the calling program module to a second subprogram included in the other program module (10:10 - 25),

an address resetting program generating means for generating an address resetting program for reading and resetting the saved data memory area address for the calling program module after the return from the second subprogram as a transfer end to the first subprogram (FIG. 3, 114 see replace and all associated text), and

an accessing program generating means for generating an accessing program for accessing a data memory area for the other program module using a relative address from the set data memory area address for the other program module when the other program module accesses the data memory area therefor included therein (FIG. 3, 104 and all associated text).

Regarding claim 2, a compiler program according to claim 1, wherein:

the computer is caused to further function as a discriminating means for discriminating whether or not to shorten a process based on a description of the other program module including the second subprogram to be called from the first subprogram included in the calling program module included in the source program (11:20 – 35, see optimization),

if the discriminating means discriminates the process not to be shortened (11:20 – 35, see no optimization),

the address saving program generating means generates the address saving program for saving the data memory area address used by the calling program module (5:37 – 55),

the address setting program generating means generates the address setting program for setting the data memory area address used by the other program module to be called by the calling program module (6:37 - 43, see setting pointer),

the transferring program generating means generates the transferring program for the transfer from the first subprogram included in the calling program module to the second subprogram included in the other program module (10:10 - 25),

the address resetting program generating means generates the address resetting program for reading and resetting the saved data memory area address for the calling program module after the return from the second subprogram as the transfer end to the first subprogram, and the accessing program generating means generates the accessing program for accessing the data memory area for the other program module using the relative address from the set data memory area address for the other program module when the other program module accesses the data memory area therefor included therein (6:37 - 43, see setting pointer), and

if the discriminating means discriminates the process to be shortened, the transferring program generating means generates the transferring program for the transfer from the first subprogram included in the calling program module to the second subprogram included in the other program module, and the accessing program generating means generates the accessing program for accessing the data memory area for the other program module using the relative address from the data memory area address for the calling program module when the other program module accesses the data memory area therefor included therein (FIG. 3, 104 and all associated text).

Regarding claim 3, a compiler pro gram according to claim 1, wherein, after the second subprogram included in the other program module is called from the first subprogram included in the calling program module included in the source program (9:20 – 30);

the address setting program generating means generates an address setting program for reading the data memory area address for the other program module from data memory area address tables for the respective program modules saved in executing units of the calling program module and setting this data memory area address (9:40 - 50), and

the accessing program generating means generates the accessing program for accessing the data memory area for the other program module using the relative address from the set data memory area address for the other program module when the other program module accesses the data memory area therefor included therein (FIG. 3, 104 and all associated text).

Regarding claim 4, a compiler program according to claim 3, wherein the computer is caused to further function as a discriminating means for discriminating whether or not to shorten a process based on a description of the other program module including the second subprogram to be called from the first subprogram included in the calling program module included in the source program (9:20 - 10:30),

if the discriminating means discriminates the process not to be shortened, after the second subprogram included in the other program module is called from the first subprogram included in the calling program module included in the source program (9:20 - 10:30),

the address setting program generating means generates the address setting program for reading the data memory area address for the other program module from the data memory area address tables for the respective program modules saved in executing units of the calling program module and setting this data memory area address (9:20 - 10:30),

and the accessing program generating means generates the accessing program for accessing the data memory area for the other program module using the relative address from the set data memory area address for the other program module when the other program module accesses the data memory area therefor included therein (9:20 - 10:30), and if the discriminating means discriminates the process to be shortened, the accessing program generating means generates the accessing program for accessing the data memory area for the other program module using the relative address from the data memory area address for the calling program module when the other program module accesses the data memory area therefor included therein (9:20 - 10:30).

Regarding claim 5, a compiler program according to claim 2, wherein the description of the other program module including the second subprogram to be called from the first subprogram included in the calling program module included in the source program is a declaration described for each second subprogram (7:55 - 67, see declarations).

Regarding claim 6, a compiler program according to claim 1, wherein the source program including the other program module including the second subprogram to be called by the calling program module includes a description for specifying which of an execution common data memory area for the other program module external variables included in the other program module commonly access upon a plurality of executions of the calling program module and an execution intrinsic data memory areas for the other program module the external variables included in the other program module commonly access every time the calling program module is executed is to be used (3:25 – 35, see selected memory address, pointer, execution and access, also 10:35 – 40 see conventional memory allocation routine).

Regarding claim 7, a compiler pro gram according to claim 6, wherein the description for specifying which of the execution common data memory area for the other program module and the execution intrinsic data memory area for the other program module is to be used is a declaration described for each external variable included in the other program module (7:55 – 67, see declarations).

Regarding claim 9, a compiler program according to claim 6, wherein, after the other program module is called from the calling program module included in the source program:

the address setting program generating means generates an address setting program for reading the execution intrinsic data memory area address for the other program module and the execution common data memory area address for the other program module commonly accessed by a plurality of executions of the calling program module from the data memory area address tables for the respective program modules saved in executing units of the calling program module and setting these data memory area addresses (3:25 – 35, see selected memory address, pointer, execution and access, also 10:35 – 40 see conventional memory allocation routine), and

the accessing program generating means generates an accessing program for accessing the execution intrinsic data memory area for the other program using a relative address from the set execution intrinsic data memory area address for the other program module when the other program module accesses the execution intrinsic data memory area address therefor included therein and accessing the execution common data memory area for the other program using a relative address from the set execution common data memory area address for the other program module when the other program module accesses the execution intrinsic data memory area address therefor included therein (3:25 – 35, see selected memory address, pointer, execution and access, also 10:35 – 40 see conventional memory allocation routine).

Regarding claim 10, a compiler program according to claim 1, wherein the address saving program generating means generates an address saving program for saving the execution common data memory area address and the execution intrinsic data memory area address for the calling program module in a stack memory prior to data necessary for the transfer from the first subprogram to the second subprogram (FIG. 3, 106 and all associated text).

Regarding claim 11, a compiler program according to of claim 1, wherein the calling program module included in the source program and the other program module including the second subprogram to be called from the first subprogram included in the calling program module are the same program module (10:20 – 30).

Regarding claims 12, a compiler program according to claim 1, wherein the calling program module includes a code area and a data area, the other program module includes a code area and a data area, and the computer is caused to further function as:

an identification-number obtaining program generating means for generating an identification-number obtaining program for obtaining an identification number for identifying the other program module upon the receipt of a call command to the other program module from the calling program module (9:45 – 50, see hashing), a leading-address obtaining program generating means for generating a leading-address obtaining program for obtaining the leading address of the data area of the other program module using the obtained identification number as an index from a table relating an identification number of the calling program module to the leading address of the data area of the calling program module and relating the identification number of the other program module to the leading address of the data area of the other program module, (8:60 – 67, see retrieval for obtaining address) and a leading-address switching program generating means for generating a leading-address switching program for switching the obtained leading address of the data area of the other program module and the leading address of the data area of the calling program module (7:55 – 65, see switching).

Regarding claims 13 and 29, a compiler program according to claim 1, wherein the calling program module includes a code area and a data area, the other program module includes a code area and a data area, and the computer is caused to further function as: an identification-number designating program generating means for generating an identification-number generating program for designating an identification number for identifying the calling program module and an identification number for identifying the other program module(9:45 – 50, see hashing), a table preparing program generating means for generating a table preparing program for preparing a table relating the designated identification number of the calling program module to the leading address of the data area of the calling program module and relating the designated identification number of the other program module to the leading address of the data area of the other program module (7:35 – 45, see segment table), an identification-number obtaining program generating means for generating an identification-number obtaining program for obtaining the identification number of the other program module upon the receipt of a call command to the other program module from the calling program module (9:45 – 55), a leading-address obtaining program generating means for generating a leading-address obtaining program for obtaining the leading address of the data area of the other program module from the prepared table using the obtained identification number as an index, and a leading-address switching program generating means for generating a leading-address switching program for switching the obtained leading address of the data area of the other program module and the leading address of the data area of the calling program module (7:35 – 65).

Regarding claims 14 and 30, a compiler program according to claim 1, wherein the computer is caused to further function as an address latching program generating means for generating an address latching program for latching an address of a variable or function defined outside the other program module, and the accessing program generating means generates an accessing program for accessing a variable defined inside the other program module using a relative address from the obtained leading address, accesses a function defined inside the other program module using a relative address from a program counter, and indirectly accessing the variable or function defined outside the other program module via the latched address after accessing the latched address using a relative address from the obtained leading address (FIG. 5 and all associated text).

Regarding claim 15, a compiler program according to claim 1, wherein the computer is caused to further function as:

an ending program generating means for generating an ending program for ending an application comprised of at least one module and being presently executed (11:10 – 15, see terminated), a compaction program generating means for generating a compaction program for compacting a memory after the application is ended, and a resuming program generating means for generating a resuming program for resuming the application from the beginning after the memory is compacted (10:3 – 10, see starting).

Regarding claim 16, a compiler program according to claim 1, wherein the computer is caused to further function as:

an ending program generating means for generating an ending program for ending an application comprised of at least one module and being presently executed, a saving program generating means for generating a saving program for saving information independent of address values of the application until the application was ended, a compaction program generating means for generating a compaction program for compacting a memory, and an executing program generating means for generating an executing program for reading the saved information and executing the application up to a state where the application was ended based on the read information after the memory is compacted (FIG. 3, 106 and all associated text).

Regarding claims 17 and 18, a compiler program according to claim 1, wherein an identifier indicating whether or not to save an address value is provided in a memory, and the computer is caused to further function as:

a memory-state saving program generating means for generating a memory state saving program for saving the state of the memory before the compaction, a compaction program generating means for generating a compaction program for compacting the memory, and a relocating program generating means for generating a relocating program for relocating the address value based on the identifier for an entry having the address value set by referring to the saved state of the memory before the compaction, after the memory is compacted (FIG. 3, 106 and all associated text).

Regarding claim 19, a compiler program according to claim 1, wherein identification numbers for identifying the modules and identifiers indicating whether to set addresses of data areas of the modules or to set addresses of code areas of the modules are provided in a memory, and the computer is caused to further function as:

a setting program generating means for generating a setting program for setting the identification number and the identifier upon substituting an address value for a pointer, a compaction program generating means for generating a compaction program for compacting the memory, and a recalculating program generating means for generating a recalculating program for recalculating the address value based on the identification number and the identifier after the memory is compacted (6:37 – 43, see setting pointer).

Regarding claim 20, a compiler program according to claim 1, wherein identification numbers for identifying the modules and identifiers indicating whether to set addresses of data areas of the modules or to set addresses of code areas of the modules are provided in a memory, and the computer is caused to further function as:

an offset-value setting program generating means for generating an offset-value setting program for setting the identification number and the identifier upon substituting an address value for a pointer and setting an offset value from the leading address of the data area or the leading address of the code area, and an actual-address converting program generating means for generating an actual-address converting program for the conversion into an actual address by adding the leading address of the data area or the leading address of the code area to the offset value upon using the pointer (6:37 – 43, see setting pointer).

Regarding claim 31, a compiler program according to claim 4, wherein the description of the other program module including the second subprogram to be called from the first subprogram included in the calling program module included in the source program is a declaration described for each second subprogram (9:20 – 30).

Allowable Subject Matter

7. Claim 21 is objected to as being dependent upon a rejected base claim, but would be allowable if rewritten in independent form including all of the limitations of the base claim and any intervening claims.

"...compaction program generating means for generating a compaction program for compacting each code area of the modules or each data area of the modules, a searching program generating means for generating a searching program for searching whether or not there is any handle indicating the code area or the data area shifted by the compaction, and a handle relocating program generating means for generating a handle relocating program for relocating the handle if the handle indicating the code area or the data area is found, wherein the compaction program generating means generates the compaction program for compacting the code areas or the data areas after all the handles are relocated..."

Correspondence Information

8. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Chuck Kendall whose telephone number is 571-272-3698. The examiner can normally be reached between Monday and Thursday, at 11:00 am - 4:300pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Dam can be reached on 571-272-3695. The fax phone number for the organization where this application or proceeding is assigned is **571-273-8300**.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Chuck O Kendall/

Primary Examiner, Art Unit 2192